Data Security

Lecture No. 6

Dr/ Roayat Ismail

Public Key Encryption

Objectives

1. Explain the basic principles behind public key cryptography .
2. Recognise the fundamental problems that need to be solved before public key cryptography can be used effectively
3. Explain the concept of a one-way function
4. Describe the RSA encryption system
5. Calculate very simple numerical examples of RSA.
6. Define Public key practical applications.

Public Key Cryptography

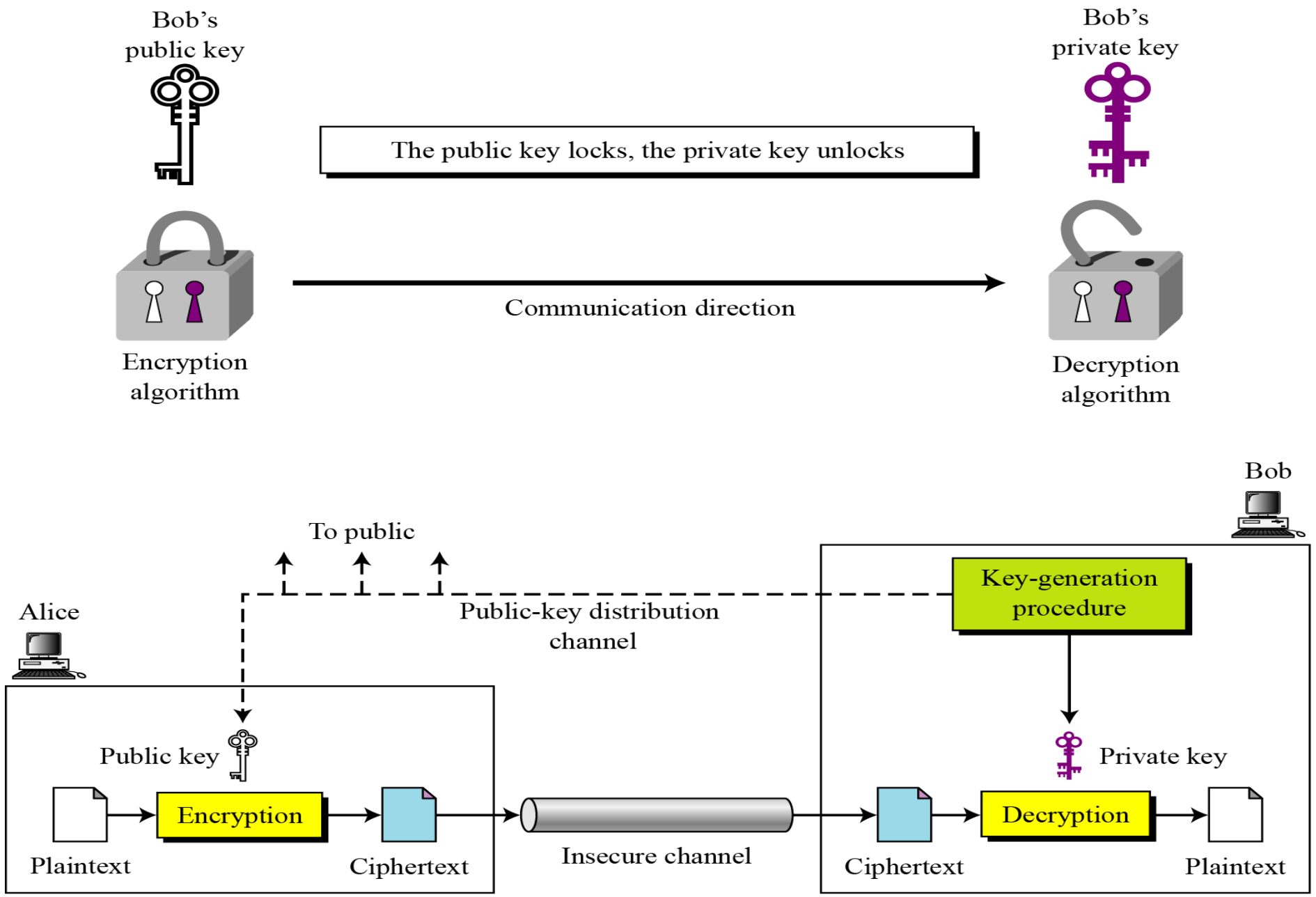
- Introduced by Diffie & Hellman in 1976.

- Features
  - Each user has two keys (a public key and a private key)
  - The algorithm is public knowledge.
  - Knowledge of the algorithm does not help an attacker.

Asymmetric-key (Public key) Encryption

The figure shows the general idea of asymmetric-key cryptography as used for confidentiality. The figure shows that, unlike symmetric-key cryptography, there are distinctive keys in asymmetric-key cryptography: a private key and a public key

If encryption and decryption are thought of as locking and unlocking padlocks with keys, then the padlock that is locked with a public key can be unlocked only with the corresponding private key.



The general idea behind asymmetric-key cryptography

Main Properties

- The keys used to encrypt and decrypt are different.
- Anyone who wants to be a receiver needs to "publish" an encryption key, which is known as the public key.
- Anyone who wants to be a receiver needs a unique decryption key, which is known as the private key.

- It should not be possible to deduce the plaintext from knowledge of the ciphertext and the public key.
- Some guarantee needs to be offered of the authenticity of a public key. Eve should not be able to advertise her public key to the community pretending that it is Bob's public key.
- Public key encryption solves the problem of symmetric key encryption (no need for secure channel for key exchange).

Design of a public key algorithm

It is based on one-way functions .

A **one-way function** is a function that is "easy" to compute and "difficult" or computationally hard to reverse.

Number Theory concepts

- Prime Numbers
  - Integers that have no positive factors except themselves and 1.
- Composite Numbers
  - Integers that have at least one factor except themselves and 1.
- Co-prime or Relatively Prime
  - Two integers a and b are co-prime if GCD(a, b)=1.
- GCD(a, b) = Largest integer that completely divides both a and b.
- Euclid's algorithm can be used to compute GCD.

More Number Theory

- Euler's Totient function
  - $\phi(n)$ = Count of numbers < n that are co-prime to n
- If n is prime
  - $\phi(n) = n-1$
- If n is composite (e.g. $n=p \cdot q$)
  - $\phi(n) = \phi(p \cdot q) = \phi(p).\phi(q) = (p-1).(q-1)$
  - p and q must be co-prime.
- Euler's Theorem
  - Given a number n, $\forall a \in \{1, 2, 3, \ldots, n-1\}$
  - GCD(a, n)=1 => $a^{\phi(n)} \bmod n = 1$

Example on one-way function is Multiplication of two primes

It is easy to take two prime numbers and multiply them together. If they are fairly small we can do this in our heads, on a piece of paper, or on a calculator. As they get bigger and bigger it is fairly easy to write a computer program to compute the product.

Multiplication of two primes is easy.

But factorization of a composite integer into its prime factors is computationally hard (intractable).

Integer Factorization is the inverse of multiplication of two prime numbers.

Integer Factorization is **believed** to be a one-way function.

We say **believed** because nobody has been able to **prove** that it is absolutely hard to factorise.

Maybe one day someone will find a way of factorising efficiently.

What will happen if someone does find an efficient way of factorising ?

Another example on one-way function is modular exponentiation

The process of **exponentiation** just means raising numbers to a power.

Raising **a** to the power **b**, normally denoted $\mathbf{a^b}$ just means multiplying **a** by itself **b** times. In other words:

$$\mathbf{a^b} = \mathbf{a} \times \mathbf{a} \times \mathbf{a} \times \ldots \times \mathbf{a}$$

Modular exponentiation means computing $\mathbf{a^b}$ modulo some other number **n**. We tend to write this as

$$\mathbf{a^b \bmod n}.$$

Modular exponentiation is “easy”.

However, given **a** and $\mathbf{a^b \bmod n}$ (when **n** is prime), calculating **b** is regarded by mathematicians as a hard problem.

This difficult problem is often referred to as the **discrete logarithm problem** (DLP).

In other words, given a number **a** and a prime number **n**, the function

$$\mathbf{f(b) = a^b \bmod n}$$

is believed to be a one-way function.

Examples of public-key encryption

1. RSA: is based on IFP
2. ElGamal and a Diffie-Hellman

   are based on DLP.

RSA

The **RSA** public key encryption algorithm was the first practical implementation of public key encryption discovered.

It remains the most used public key encryption algorithm today.

It is named after the three researchers Ron Rivest, Adi Shamir and Len Adleman who first published it.

RSA

- Invented by Rivest, Shamir & Adleman in 1978.
- Public key cryptosystem based on the Integer Factorization problem.
- Very Popular
- One of the first to support Digital Signatures.

RSA – Key Generation

- Every user
  - Picks two large random prime numbers (p, q)
  - By "large" we typically mean at least 512 bits.
  - Computes $n = p \cdot q$
  - Computes $\phi(n) = (p-1).(q-1)$
  - Picks a random integer e
    - $1 < e < \phi(n)$
    - $GCD(\phi(n), e) = 1$

[no common factors between e and (p-1)(q-1) except 1].

  - Computes $d = e^{-1} \bmod \phi(n)$
- Public Key = (n, e)
- Secret Key = $(\phi(n), p, q, d)$

Encryption/Decryption

- Encryption (raise M to the e^{th} power in mod n)
  - $C = M^e \bmod n$
- Decryption (raise C to the d^{th} power in mod n)
  - $M = C^d \bmod n$
- Works because e & d are inverses
  - $e.d = 1 \bmod \phi(n)$
  - $(M^e)^d \bmod n = M^1 \bmod n$

Breaking RSA

- Public knowledge = (n, e)
- Secret knowledge = (ϕ(n),p , q , d)

- d cannot be computed without knowing ϕ(n).
  - Recall that $d = e^{-1} \bmod \phi(n)$

- An attacker must compute ϕ(n) given only n.
  - Need to factorize n into its prime factors, which is IFP.

Integer Factorization

- Stated as a search problem
  - Given an integer n, find its prime factors.
- Brute-force approach:
- RSA typically uses n of 1024 bit to be secure.

Choosing e

Let's consider p =3 and q=7. What choices of e are acceptable?

In this case (p -1)(q -1) = 2 x 6 = 12. Any suitable choice of e must have the property that [no common factors between e and 12 except 1].

Let's just try them all out:

e = 2: this is no good, since 2 divides both e and 12. In fact this will be true for all multiples of 2 as well, so **e=4**, **e=6**, **e=8** and **e=10** are also not possible.

e=3: this is no good, since 3 divides both e and 12. In fact this will be true for all multiples of 3 as well, so **e=6** and **e=9** are also not possible.

The remaining choices are **e=5, e=7** and **e=11**. Since in each case there is no number that divides into them and 12 other than 1, all these choices of e are possible.

Setting up RSA: example-1

Step 1: Let p = 3 and q =5. Thus n = 3x 5=15

Step 2: : (p-1) x (q-1) = 2 x 4=8

Select e =3

Step 3: Publish (n,e) = (15,3)

Step 4: Use the Euclidean Algorithm to compute the modular inverse of 3 mod 8. The result is

$d = 3^{-1} \bmod 8 = 3^{8-1} \bmod 8 = 3^7 \bmod 8 = 3$

<< Check: 3 x 3= 9 = 1(mod 8) >>

Public key is (15,3)

Private key is 3

Setting up RSA: example-2

Step 1: Let p = **47** and q = **59**. Thus n = **47** x **59** = **2773**

Step 2: : (p-1) x (q-1) = **46** x **58** = **2668**
Select e = **17**

Step 3: Publish (n,e) = (**2773**, **17**)

Step 4:
Use the Euclidean Algorithm to compute the modular inverse of **17** modulo **2668**. The result is d = **157**

<< Check: **17** x **157** = 2669 = 1(mod **2668**) >>

Public key is (**2773**,**17**)
Private key is **157**

Encryption and decryption: example

Public key is (**2773**,**17**)
Private key is **157**

Plaintext block represented as a number: **M = 31**

Encryption using Public Key: $C = 31^{17} \pmod{2773}$

$= 587$

Decryption using Private Key: $M = 587^{157} \pmod{2773}$

$= 31$

Length of an RSA modulus (n)

- Factorization algorithm called GNFS. In Aug-99 succeed in factoring n of 130 decimal digits (512) bit

– biggest improvement comes from improved algorithm.

– RSA is secure with bit length of n =1024 bit

Length of an RSA modulus

It is hard to compare the equivalent security parameters for symmetric key cipher systems and RSA, however it is roughly believed that factorising a 512 bit number is about as hard as searching for a 56 bit symmetric key.

Assignments

1. Perform encryption and decryption using RSA algorithm, as in Figure 1, for the following:

   ① p = 3; q = 11, e = 7; M = 5

   ② p = 5; q = 11, e = 3; M = 9

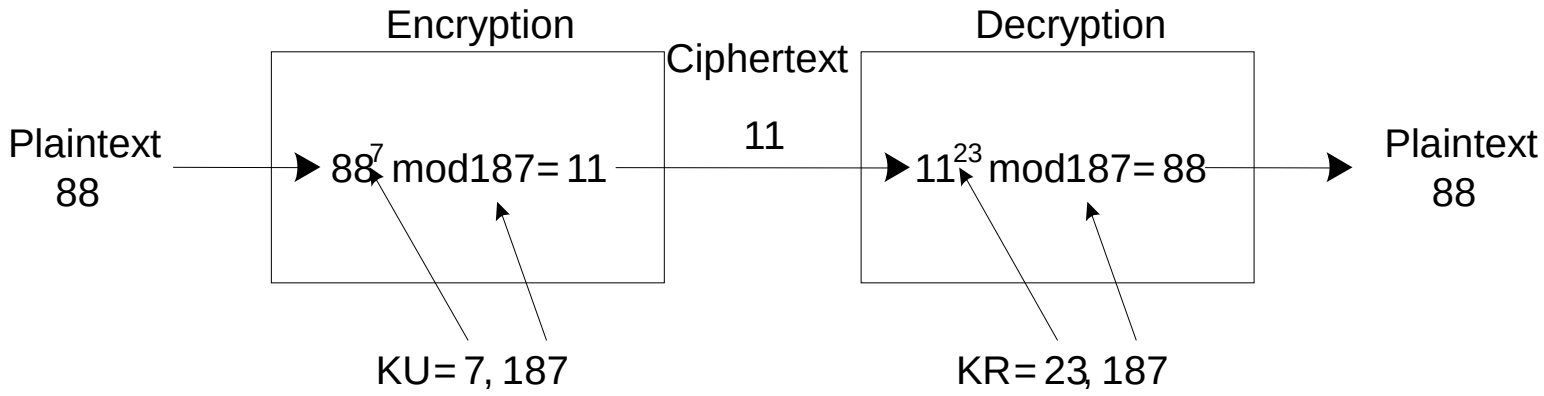


Figure 1. Example of RSA Algorithm

2. In a public-key system using RSA, you intercept the ciphertext C = 10 sent to a user whose public key is e = 5, n = 35. What is the plaintext M?

Public key systems in practice

- Public key cipher systems led to mini revolution in cryptography in the mid 1970's, with a further boom in interest since the development of the Internet in the 1990's.

- Public key cipher systems are only likely to grow in importance in the coming years.

Applications of public key encryption

- One of the major applications of public key cipher systems is for digital signatures.
- A second major application of public key cipher systems is to distribute and transfer symmetric keys around a network, thus presenting public key cipher systems as a useful enabler for faster symmetric cipher systems.
- The big problem of public key encryption is authenticating of public keys.

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication and non-repudiation)
  - **key exchange** (of session symmetric keys encryption)
- some algorithms are suitable for all uses, others are specific to one.